이제는 패턴 추출을 넘어서야 할 때

이희원

2022년 11월 15일

요 약

현재 개발되는 인공지능의 모델은 대부분 인공 신경망 및 이를 응용한 딥러닝 모델이다. 이를 이용해 우리는 프로그램 합성, 문학 창작 인공지능 등을 개발해내는 쾌거를 일궜다. 그러나 우리는 패턴 추출 인공지능 모델의 한계에 봉착했다. 실무 환경에서 이러한 프로그램 합성 인공지능으로 인간 개발자를 대체하기에는, 인공지능이 합성하는 프로그램에서 보안에 위협이 가고 정확도를 떨어뜨리는 오류가 너무 많이 발견된다. 이는 인공지능이 프로그래밍 언어의 의미를 이해하지 못한 채로 패턴만을 적용하기 때문이며, 이를 해결하기 위해서는 기존의 기호주의적 프로그램 분석 기법이 사용되어야 한다. 기호주의적 인공지능 등의 연구를 통해 패턴 추출을 넘어서, 맥락 이해 등 다음 단계의 인공지능으로의 발전을 목표해야 한다.

과연 컴퓨터가 소프트웨어 개발을 온전히 짊어질 수 있을까? 현대적인 컴퓨터를 설계한 앨런 튜링(Alan Turing)이 1946년에 제시했던 이 질문은 그 출생 이후로 끊임없는 전산학 연구의 성배(holy grail)로 취급되어 왔다. 그리고 이 질문은 작년 깃허브(Github)에서 제시한 코드 자동 완성 인공지능, 코파일럿(Copilot)의 출시로 새로운 국면을 맞이했다. 개발자들은 단순하고 기계적인 코드 작성에 쓰이는 시간이 코파일럿을 이용함으로써 절반 가량 단축되었다고 말했고, 몇몇 전문가들은 약 10년 정도 이후에는 소프트웨어 개발을 위해서 프로그래밍 언어를 공부할 필요가 없을 것이라는 낙관적인 예측을 던지기도 했다. 하지만 정말 인공지능(AI)이 소프트웨어 개발자들을 대체할 수 있을까? 이에 답하기 위해서는 코드 작성 AI가 기작하는 방식에 대해 고찰할 필요가 있다.

현재 자동 프로그램 합성에 쓰이는 인공 신경망(neural network) 기술은 패턴 추출 이상의 작업을 수행하지 못한다. 인공 신경망은 생명체의 뇌에서 정보를 처리하는 방식을 모방하여 다양한 문제의 해법을 학습할 수 있는 기계학습(machine learning)의 한 모델이다. 신경망을 겹겹이 쌓은 딥러닝(Deep learning) 모델을 비롯한 인공 신경망은 앞서 말한 코파일럿, 특정 주제에 대한 문서를 작성하는 자기회귀 언어 모델(GPT-3) 등에서 뛰어난 성능과 속도를 보였다. 그러나, 인공 신경망의 기능은 학습한 데이터로부터 패턴을 인식하고 새로운 데이터에 이를 적용하는 것이다. 이러한 접근의 한계는 다양한 방식으로 모습으로 드러나고 있는데, 이중 프로그램 합성에 있어 가장 심각한 점은 단순한 패턴을 넘어 프로그래밍 언어의 의미(sematics)를 전혀 고려하지 못한다는 점이다.

프로그래밍 언어의 의미를 이해하지 못한다는 점은 전산학 연구에 있어 치명적인 문제이다. 흔히 불투명 문제(blackbox problem)라 불리는 것도 이에서 기인한다고 볼 수 있다. 말했다시피 현재의 프로그램 합성 인공지능은 제공된 데이터, 즉 사람들이 작성한 코드에서 패턴을 발견하고 적용하는 형태이다. 즉, 사람이 흔히 만드는 오류(bug) 역시 학습되어 합성한 프로그램에 나타날 수도 있다는 것이다. 실제로 뉴욕대학교에서 실

시한 연구에 따르면 코파일럿이 작성한 코드 중 4할 정도가 보안에 위협이 가는 오류를 포함하고 있고, 데이터 과학자 알렉스 나카(Alex Naka)는 코파일럿을 이용할 경우 본인의 직무가 코드 작성에서 합성된 코드 속 오류 수정으로 이전한 것 같다는 응답을 보였다. 또한, 이러한 버그로 인해 합성된 프로그램의 정확성 자체에 문제가 가는 점도 무시할 수 없다. 무려 구글의 딥마인드(DeepMind)에서 개발한 코드 합성기 알파코드(AlphaCode)는 딥러닝과 거대 언어 모델(large language model) 등의 최첨단 기술을 이용해 자연어로부터 코드를 생성하는 인공지능이다. 그러나 이러한 이름표가 무색하게도, 이전에 인간 개발자들을 대상으로 진행되었던 프로그래밍 대회의 문제를 풀어본 결과는 상위 54%, 평균보다 살짝 낮은 등수에 그쳤다. 마이크로소프트에서 개발한 티코더(TiCoder) 코드 합성기의 경우 85%의 정확도를 보여주지만, 이는 인간 개발자가 지속적으로 프로그램의 성능에 대해 피드백을 제공했기에 가능한 수치이다. 인간 개발자 없이 패턴 인식만으로 합성한 코드의 정확도는 48%에 지나지 않는다. 이처럼 실무 환경에서 치명적일 수 있는 버그가 생기는 이유는 인공지능이 프로그래밍 언어를 제대로 이해하지 못했기 때문에, 해당 코드를 실행함으로써 일어날 기작을 전혀 예상하지 못했기 때문이라 정리할 수 있다.

이러한 벽을 넘기 위한 시도로, 인공 신경망과 기존 프로그래밍 언어 연구 방식을 결합하는 기호주의 인공지능(Symbolic AI)가 제시되었다. 이전의 전산학 연구는 논리학을 기반으로 하여, 프로그램의 기작을 검증하고 원하는 코드를 합성할 수 있는 논리적인 알고리즘을 연구하는 방식으로 진행되어 왔다. 실제 데이터의 값들이 아니라 논리 기호를 기반으로, 엄밀하게 정확성을 검증할 수 있다는 점에서 "기호주의"라는 명칭이 붙은 것이다. 이 말에서 알 수 있듯이 기호주의적 접근의 최대 장점은 엄밀성이다. 어떤 프로그램이 정확히 작동함을 논리적으로 증명할 수 있다는 점은, 본질적인 불투명함 속에서 연구되는 기계학습이 넘볼 수 없는 강점이다. 하지만 이러한 기호주의적 접근으로 개발한 프로그램의 경우 속도가 매우 느리고, 제약 조건이 많이 주어지는 제한적인 상황에서만 적용할 수 있었다. 그렇기 때문에 엄밀함보다는 범용성과 성능에 집중한 기계학습 연구가 활발히 진행되었던 것이다. 그러나 이제 우리는 각 접근법을 따로 고려해서는 원하는 문제를 풀 수 없는 처지에 도달했다. 합성된 프로그램의 정확성을 보장할 수 있으면서도 넓은 스펙트럼의 문제들을 해결할 수 있는 강력한 인공지능이 필요하기 때문에, 우리는 새로운 방식을 연구해야 한다. 그 일례로 제시된 것이 앞서 말한 기호주의 인공지능이다. 인공 신경망의 속도와 정확성, 그리고 논리적 접근의 엄밀성을 동시에 취할 수 있게끔 두 방식을 적절히 혼합하여 인공지능을 학습시키는 것이다. 이러한 기호주의 인공지능은 여러 형태를 가질 수 있다. 인공 신경망을 논리적인 기호나 제약 조건으로 학습시키거나, 신경망의 학습 결과를 논리적으로 분석하는 등의 방식이 연구되고 있다. 이러한 방식이 최종적으로 추구하는 바는 인공지능의 기작을 논리적으로 기술하는, 즉 인공지능의 의미(semantics)를 정립하는 것이다. 이를 통해 우리는 프로그램 합성 뿐 아니라 다양한 문제를 현상론적이 아닌 환원주의적인 방식으로 해결할 수 있을 것이다.

우리는 이제 패턴 추출 이상의 기능을 컴퓨터가 수행할 수 있게끔 노력하는 방향으로 패러다임을 전환해야 한다. 그동안 우리는 인공 신경망을 비롯한 패턴 추출 모델을 통해 인공지능 분야에서 획기적인 발전을 일궈낼 수 있었다. 튜링에 시대에는 상상만 할 수 있었던 코드 합성, 문학 창작 알고리즘 등을 개발할 수 있었다. 하지만 우리는 이 업적을 자랑스러워하되, 이에 만족해서는 안 된다. 원리를 이해하지 못하고 주어진 데이터를 모방하는 데에 그치는 지금의 인공지능은 실무 환경에서는 일어나서는 안되는, 보안 및 정확성에 영향을 미치는 오류를 범하고 있으며, 인간의 개입 없이는 이를 인지하지도, 수정하지도 못한다. 앞으로의 전산학 연구는 기호주의 인공지능 등의 모델을 연구함으로써, 그 기작과 의미가 논리적으로 명확히 기술되는 인공지능 개발을 목표로 해야 한다. 단순히 패턴을 인식하고 적용하는 데에 그치지 말고, 이야기의 맥락 파악 및 데이터 속의 인과관계 분석 등이 가능한 인공지능을 만드는 것이 앞으로의 목표라 할 수 있겠다.